Here’s a roadmap to learn the basics of any programming language (Python, Java, JavaScript, C++, etc.) within a week, along with 100 practice exercises. The plan covers key topics every day, allowing you to build a strong foundation and practice through exercises.

**Day 1: Introduction and Setup**

**Learning Goals:**

* **Understanding the language**: History, uses, syntax, basic setup.
* **Programming Environment**: Set up IDE (VSCode, PyCharm, etc.).
* **Hello World**: Write your first program.
* **Basic Syntax**: Comments, printing output, taking input.

**Topics:**

* Variables and Data Types
* Input/Output
* Basic Operators (Arithmetic, Comparison)

**Exercises:**

1. Write a program to display "Hello World".
2. Create a program that asks the user’s name and greets them.
3. Write a program to add two numbers.
4. Create a program to find the remainder of two numbers.
5. Write a program to swap two variables.
6. Write a program to calculate the area of a rectangle.
7. Create a program to convert Fahrenheit to Celsius.
8. Write a program to check if a number is positive or negative.
9. Write a program to calculate the sum of three numbers.
10. Write a program to find the average of five numbers.

**Day 2: Control Flow and Loops**

**Learning Goals:**

* **Conditionals**: if-else statements, switch (if supported).
* **Loops**: while, for, do-while loops.

**Topics:**

* Conditional Statements (if, elif/else)
* Loops (for, while)
* Break, Continue

**Exercises:**

1. Write a program to find if a number is even or odd.
2. Create a program to find the largest of three numbers.
3. Write a program to calculate the factorial of a number.
4. Create a program to check if a number is prime.
5. Write a program to display the multiplication table of a number.
6. Write a program to find the sum of first n natural numbers.
7. Create a program to reverse a number.
8. Write a program to find the greatest common divisor (GCD) of two numbers.
9. Create a program to count the number of digits in a number.
10. Write a program to check if a number is a palindrome.

**Day 3: Functions and Modularity**

**Learning Goals:**

* **Functions**: Define and use functions.
* **Return Values**: Returning values from functions.
* **Parameters**: Passing arguments to functions (default, optional).

**Topics:**

* Defining Functions
* Function Parameters and Arguments
* Return Values

**Exercises:**

1. Write a function to find the maximum of two numbers.
2. Create a function to return the cube of a number.
3. Write a function to calculate the area of a circle.
4. Create a function that checks if a string is a palindrome.
5. Write a function to sum an array of numbers.
6. Create a function to check if a number is prime.
7. Write a function to calculate the power of a number (without using pow).
8. Create a function to find the least common multiple (LCM) of two numbers.
9. Write a recursive function to calculate the Fibonacci sequence.
10. Write a function that converts a string to uppercase.

**Day 4: Data Structures (Arrays, Lists, Strings)**

**Learning Goals:**

* **Arrays/Lists**: Declaration, initialization, and manipulation.
* **String Manipulation**: Operations on strings.

**Topics:**

* Arrays/Lists
* String Manipulation
* Looping Through Arrays/Lists

**Exercises:**

1. Write a program to reverse an array.
2. Create a program to sort an array in ascending order.
3. Write a program to find the largest element in an array.
4. Create a program to count the number of vowels in a string.
5. Write a program to concatenate two strings.
6. Create a program to find the frequency of a character in a string.
7. Write a program to remove all spaces from a string.
8. Create a program to find the second largest element in an array.
9. Write a program to check if a string is an anagram.
10. Create a program to split a string into words.

**Day 5: Object-Oriented Programming (OOP)**

**Learning Goals:**

* **Classes and Objects**: Define classes and create objects.
* **Encapsulation**: Private and public access.
* **Methods**: Define methods inside classes.

**Topics:**

* Defining Classes
* Objects and Constructors
* Methods and Attributes

**Exercises:**

1. Write a class to represent a bank account.
2. Create a class to represent a student with attributes like name and roll number.
3. Write a class to represent a rectangle with methods to calculate the area and perimeter.
4. Create a class for a basic calculator that can add, subtract, multiply, and divide.
5. Write a class to represent a car with attributes like make, model, and speed.
6. Create a class that manages a to-do list.
7. Write a class that tracks the number of instances created.
8. Create a class for an employee with methods to calculate annual salary.
9. Write a class that checks whether a string is a palindrome.
10. Create a class for a library book with methods to check availability.

**Day 6: File Handling and Exceptions**

**Learning Goals:**

* **File Operations**: Reading from and writing to files.
* **Exception Handling**: Try-except/try-catch blocks.

**Topics:**

* File Reading/Writing
* Exception Handling

**Exercises:**

1. Write a program to read from a text file and print its contents.
2. Create a program to write a list of numbers to a file.
3. Write a program to count the number of lines in a file.
4. Create a program to append text to an existing file.
5. Write a program to read a CSV file and print each row.
6. Create a program to write user input into a file until they type "exit".
7. Write a program that catches division by zero exceptions.
8. Create a program that raises an exception if an invalid number is entered.
9. Write a program that handles file not found exceptions.
10. Create a program to log error messages to a file.

**Day 7: Final Practice and Mini Project**

**Learning Goals:**

* **Integrating Concepts**: Combine what you’ve learned into a cohesive project.

**Mini Project Ideas:**

* **Simple Calculator**: Implement a fully functional calculator.
* **Todo List Manager**: Create a CLI todo list application.
* **Number Guessing Game**: Build an interactive number guessing game.
* **Basic Bank System**: Create a program that simulates bank transactions.

**Exercises:**

1. Create a number guessing game.
2. Write a program that manages a shopping list.
3. Create a program to simulate a basic ATM withdrawal system.
4. Write a program to track your daily expenses.
5. Create a simple quiz game.
6. Write a program that manages a phonebook.
7. Create a simple voting system.
8. Write a program to calculate simple and compound interest.
9. Create a program that keeps a log of weather updates.
10. Write a program that encrypts and decrypts text.

**Additional Practice Questions (Day 7)**

1. Write a program to find the sum of all elements in a list.
2. Create a program to merge two sorted arrays.
3. Write a program to remove duplicates from a list.
4. Create a program to rotate an array.
5. Write a program to find the longest word in a string.
6. Create a program to find the most frequent element in an array.
7. Write a program to count the occurrences of each word in a string.
8. Create a program to transpose a matrix.
9. Write a program to check if a string is a valid email.
10. Create a program to implement bubble sort.
11. Write a program to implement binary search.
12. Create a program to find the common elements between two lists.
13. Write a program to generate a random password.
14. Create a program to calculate the median of a list of numbers.
15. Write a program to flatten a nested list.
16. Create a program to implement selection sort.
17. Write a program to find the intersection of two arrays.
18. Create a program to simulate rolling a dice.
19. Write a program to find the first non-repeating character in a string.
20. Create a program to sort a list of dictionaries by a key.
21. Write a program to remove duplicates from a string.
22. Create a program to check if a number is a perfect square.
23. Write a program to reverse the order of words in a sentence.
24. Create a program to implement the merge sort algorithm.
25. Write a program to shuffle a list of elements.
26. Create a program to find the longest palindromic substring.
27. Write a program to remove vowels from a string.
28. Create a program to calculate the number of days between two dates.
29. Write a program to find the missing number in a list of consecutive numbers.
30. Create a program to implement quicksort.

By the end of the week, you will have a strong understanding of the basic concepts of programming and solid practice through these 100 exercises. You can extend this plan to deepen your knowledge by working on larger projects and learning advanced topics later on.